CFD CTA Technology Improvement Projects
In
Grid Generation

Bharat K. Soni
Hugh Thornburg
Surya Dinavahi
David Thompson
Roy Koomullil
NSF Engineering Research Center
Mississippi State University
Mississippi State, MS 39762

ABSTRACT

The progress realized in CFD CTA technology improvement projects associated with grid generation is described in this paper. GGLiB (a geometry and grid library), algorithms to facilitate grid automation, and development of grid adaptation schemes with an associated feature detection module are emphasized. The software system, PMAG (Parallel Multiblock Adaptive Grid system) is also described in detail. Computational examples are presented to demonstrate the successes of these methodologies.

INTRODUCTION

With the advent and development of high performance computers, pertinent software and numerical algorithms, Computational Field Simulation (CFS) is rapidly emerging as an essential analysis tool for science and engineering problems. The availability of previously unimaginable computational power and improved numerical algorithms has fundamentally changed the way underlying principles of science and engineering are applied to research, design and development in current engineering practice. For example, Computational Fluid Dynamics (CFD) techniques, traditionally used on fluid mechanics problems involving aerodynamics, hydrodynamics, and heat and mass transfer, are now being explored for electromagnetic, bio-engineering, and other physical field problems. The reduction in design time in both the automotive and aircraft industry is evidence of the importance of computational simulation to industry. Lockheed Martin, for example, is now integrating computational simulation and modeling tools to create a virtual design and manufacturing environment for the Joint Strike Fighter and the F-22 fighter plane. Integration of computational simulation results with test data for improved analysis and ultimate design related data quality is of key importance to future technology.

The PET component of the MSRC is designed to improve the state-of-the-practice in analysis, simulation, test, and design utilizing high performance computing resources. The mission of the CFD CTA PET program at the MSRC is to improve the capability to efficiently perform CFD simulations of importance to DOD in a HPC environment. At the same time, a significant improvement in DOD expertise in HPC, with extensive awareness of CFD in the HPC environment, should be achieved improving DOD collaborations by significantly reducing technology transfer time from research and development communities to CFD practitioners (the REAL users). The overall CFD CTA mission at ARL and ASC is to play a key role in accomplishing the stated goals working side-by-side with CTA teams at all MSRCs.

BACKGROUND

The construction of a grid on which to represent and solve the discrete approximates to the field equations is an essential element of the computational simulations for problems of interest in engineering. The grid (mesh) generated must be boundary-conforming for the complete region of interest, and must exhibit point densities appropriate for the resolution of relevant physics on widely disparate length scales. The grid employed can have a profound influence on the accuracy and economy of the overall simulation. In general, the discretization of the governing equations usually depends on the grid and solution strategy (finite difference, finite volume, finite element, spectral, etc.) under consideration. The grid can be classified as cartesian, multiblock structured with attached or overlapping zones, multiblock structured with arbitrarily overlaid (chimera) blocks, unstructured, hybrid/generalized or gridless.

A pictorial vision chart representative of the distinct issues -- past, present and future states, as well as ultimate goals -- is shown in Figure 1. The main concern, however, is response time. In general, only for moderately complex configurations can the geometry be prepared and the grid be generated in a timely enough manner to fulfill industrial needs. The ultimate industrial goal is to perform the grid generation for complex design applications in one hour and the entire simulation involving complicated physics in one day. The response time chart (Figure 1) represents the average time required to perform grid generation and associated sensitivity analysis (grid generation with minor design perturbations) and industrial expectations. In general, 80 - 90% of the grid generation labor time is usually spent on the geometry preparation and surface grid generation.

In most applications these surfaces, representative of solid components, are defined in the CAD/CAM systems as a composite of explicit or implicit analytical entities, semi-analytic parametric splines, and/or a sculptured discrete set of points. The standard common interface for geometry exchange is IGES (International Graphics Exchange Standard) which is based on curve and surface definitions of the geometric entities. These entities are not suitable for the treatment of trimmed surfaces which appear frequently in CAD representations. Also, there are no standards for the tolerances and numerical approximations required for various numerical. As a result, there exists a multimillion-dollar industry developing software just to exchange IGES data from one CAD system to

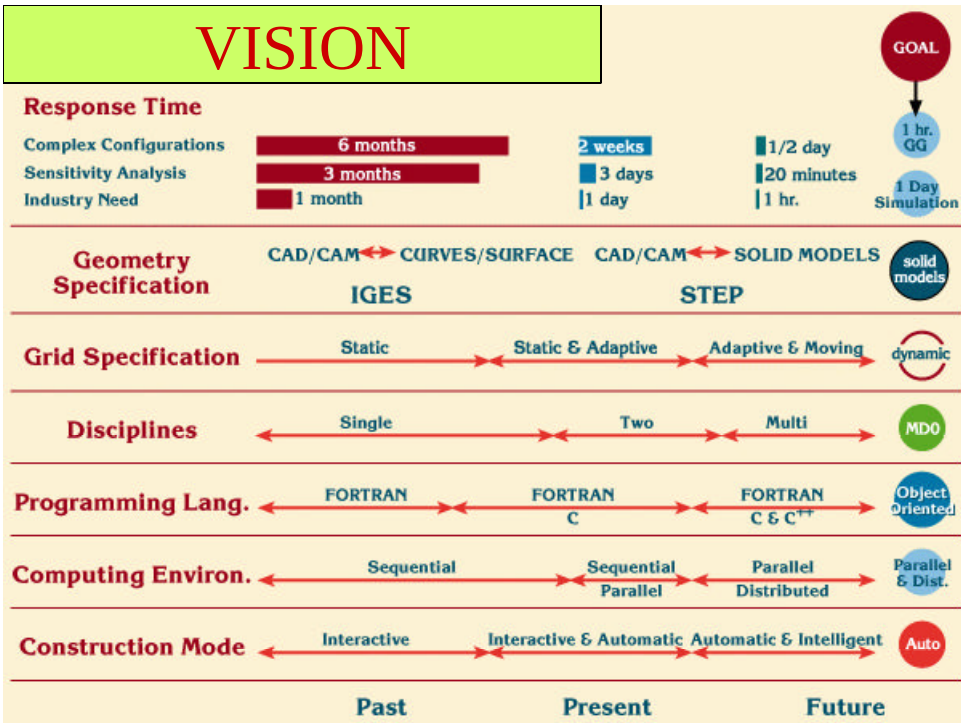


Figure 1 Vision: Past, Present, and Future

another. Also, due to different tolerance requirements, the resulting geometric description may include gaps, overlaps, and bad elements. Better methodologies are needed to repair CAD geometries and more efficient geometry and grid interfaces are needed. The CAD industry, however, is beginning to use solid modeling based geometric entities. Consequently, a new international standard, STEP (Standard for the Exchange of Product Data), is under development. The STEP is not only a geometric exchange standard. It can also be used throughout the product life cycle, including design, analysis, manufacturing, support and maintenance. The ultimate goal should be to develop grid technology based on solid models with a robust and efficient interface to STEP.

Complex simulation problems typically involve multiple scales in both the temporal and spatial domains. These simulations are performed using sophisticated spatial and temporal discretization techniques that involve finite element, finite difference or finite volume methods. It is well known that adaptive meshes should be used to obtain accurate solutions with a reasonable number of degrees of freedom. Thus, in view of affordability and accuracy requirements, it is very important to develop quality grids based on field characteristics (adaptive grids) and/or based on the movement of geometrical components in the field (moving grids). There is also an increasing demand for dynamic (adaptive/moving) grids. The dynamic grid algorithms, at present, are limited to simple configurations. Techniques are needed to enhance the applicability of adaptive schemes for complex configurations. This task requires computer representation of the geometrical configuration and intimate coupling with the grid generation software. It continues to be the pacing item in the application of computational simulation to engineering analysis and

industrial design -- requiring too much labor to produce the geometry/grid for new or modified configurations, and thus significantly delaying and lengthening the design process.

Due to reasons listed above, effort is concentrated on technology improvement projects in the area of geometry and grid generation. The progress realized in the development of CAD to grid interface treatment, algorithms for automating the grid generation, and adaptive grid generation is presented in this paper.

Discussions of GGLiB (a geometry and grid library), efforts to automate grid generation, research in the area of grid adaptation, and the development of the parallel multiblock adaptive grid system (PMAG) follow.

GGLiB

The surface definition of solid geometrical for the geometry under plays a very crucial role in the efficiency and accuracy of the overall grid generation. Various software tools, libraries, and new methodologies to treat geometrical entities and associated transformations required for analysis have been developed. In particular, the public domain libraries, DT_NURBS developed by Boeing under sponsorship of the Navy, the public domain software CAGI developed at MSU under the sponsorship of NASA, the commercial systems GRIDGEN, ICEM, CFD-GEOM, and others are based on the NURBS as the geometry engine. The development of GGLiB, a geometry and grid generation software library is designed to improve the state-of-the-practice by acquiring and enhancing or developing appropriate methodologies/modules.

There are two distinct approaches to integrate geometry and grid generation currently utilized. In the first approach, a loosely coupled module is developed by efficiently utilizing the geometry modeling and manipulation capability of the CAD system. This is accomplished via user function APIs to directly access the geometry engine of the CAD system. This approach has been successfully applied by Haimes[ ] in the development of the system, “Capri.” The second approach calls for the development of a stand-alone system to manipulate/generate/enhance geometry information from the IGES file. To this end, the development of the GGLiB[ ] is undertaken. GGLiB offers a transparent API access to widely utilized geometry and grid functionsto application software written in FORTRAN, C, C++, and JAVA. A pictorial diagram depicting GGLiB is displayed in Figure 2.

GGLiB assumes that geometrical entities are described by NURBS. Functions to tranform commonly used entities into NURBS, evaluations, transformations, and projections along with an IGES reader (which reads IGES file and converts all entities into NURBS) are accessible in this library. All functions available in the CAGI[31] system are included in GGLiB. Detailed discussions and information can be found at

.

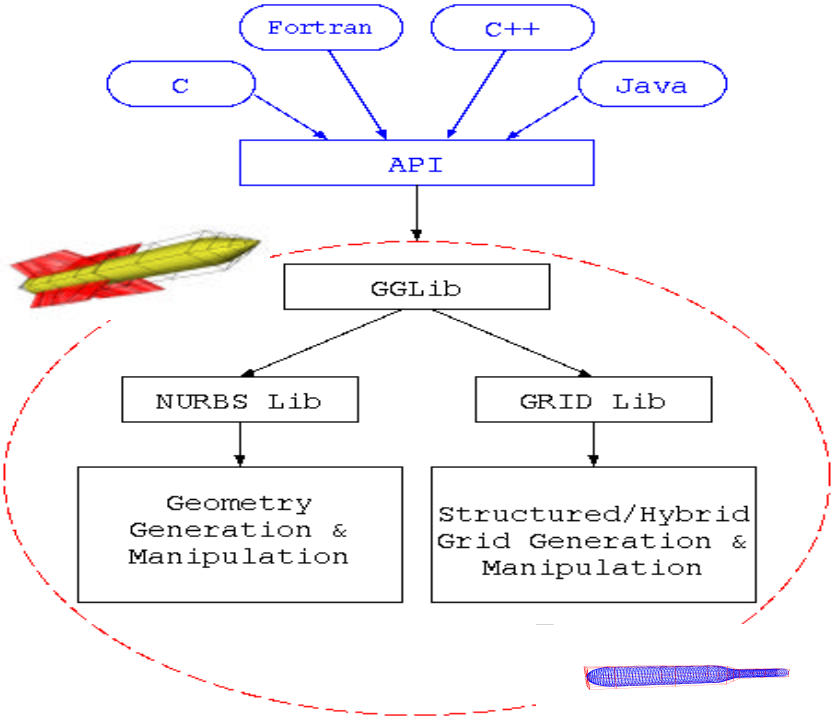


Figure 2 Structure of GGLiB

http://www.erc.msstate.edu/~bsoni/GGLiB. We expect to release GGLiB version 1.0 to DOD community at large in September'99

GRID GENERATION AUTOMATION

Two of the most commonly used approaches for grid generation are structured multiblock and unstructured grids. Each of them has it's own advantages and disadvantages. Structured grid based flow solvers are more efficient and can handle very high aspect ratio cells. But, the man-hours to generate a good quality grid are much higher compared to unstructured grids. There are accuracy issues that have to be addressed for the Navier-Stokes calculations on highly stretched unstructured grids. Recently, hybrid or generalized element grid generation strategy has been developed to combine the attractive features of structured and unstructured grids

Hybrid Grid Generation

The hybrid grids developed as part of this effort consist of a structured grid in part of the physical domain and an unstructured grid in the rest of the field. In general, a hybrid grid can be defined as an agglomeration of cells with different number of sides. This necessitates the hybrid grid generation process to be a combination of structured and unstructured grid methodologies. Good quality structured grids in the boundary layer are generated by an advancing layer type of marching together with a local elliptic solver. The overlapped structured grid from different solid bodies in the domain is trimmed by

comparing the aspect ratios of the cells. The rest of the domain is filled with unstructured grids.

Generation of Structured Grid

An advancing front scheme is utilized to generate the structured grid near no-slip boundaries (Huang [1]). Marching fronts are started from the different entities in the field. Each sculptured body surface is defined such that significant features of the body are preserved. The normal to the body at each discrete point is calculated algebraically by averaging the normals of the line segments sharing the point. The location of the point in the next layer is evaluated by utilizing a user-specified distance along the normal. Once the second layer of points is determined for all boundary points, the second layer is taken as the new marching front. A third layer is generated from the previously generated grid line using the procedure outlined above. An elliptic solver is applied to these three grid lines to smooth grid lines and avoid grid crossing. This step is important for geometries having concave and convex surfaces. The smoothing helps ensure a smooth turning of the surface normals for concave surfaces. After the application of the elliptic solver, the second grid line is taken as the new front and the process is repeated until the required number of structured grid layers in the boundary layer is reached. Since this is a marching scheme and the application of the elliptic solver is local, it makes the overall process fast and robust.

Trimming of Structured Grid

The structured grids in the vicinity of the solid bodies in the domain are generated independently. These structured grids are trimmed based on the aspect ratio of the cells and the overlapping of the component grids (Huang [1]). All cells that have an aspect ratio less than unity are removed. After clearing the cells with aspect ratio less than one and the overlapping cells, the points which are located near the cleared gap region are connected to form a closed loop. The closed outer boundary loops for the component grids together with the global outer boundary information are supplied as boundaries for the unstructured grid generation.

Sometimes two component grids may be close enough so that there is not enough space for growing the high aspect ratio cells to unity before the trimming. In this case, the boundary points are refined to get a smooth transition of the structured grid to unstructured grid.

Generation of Unstructured Part

After trimming, part of the physical domain is filled with non-overlapping structured grids and the rest of the domain is left empty. This empty region is then filled with triangular cells. The unstructured grid is generated using Delaunay triangulation (Weatherill [2]). For Delaunay triangulation the boundaries are formed by the outer boundaries of trimmed structured grids and the outer boundary of the physical domain.

As the final step of the hybrid grid generation, the trimmed structured grid and the unstructured grid are connected to form a single grid. The connectivity table is also updated with the new node numbers.

An example of hybrid grid generated using this approach is shown in Figures 3 and 4. Figure 3 shows a close-up view of the trimmed structured grid near the leading edge of an

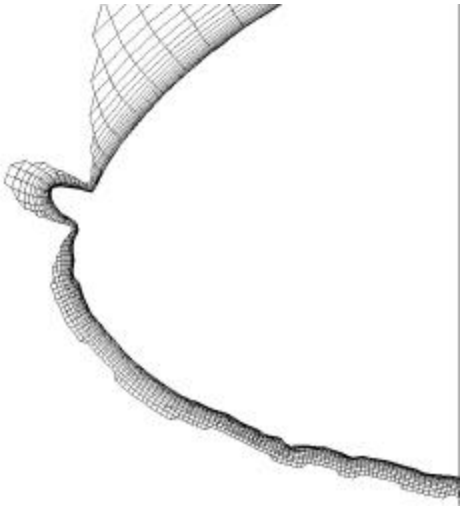


Figure 3 Trimmed Structured Grid Near the Leading Edge

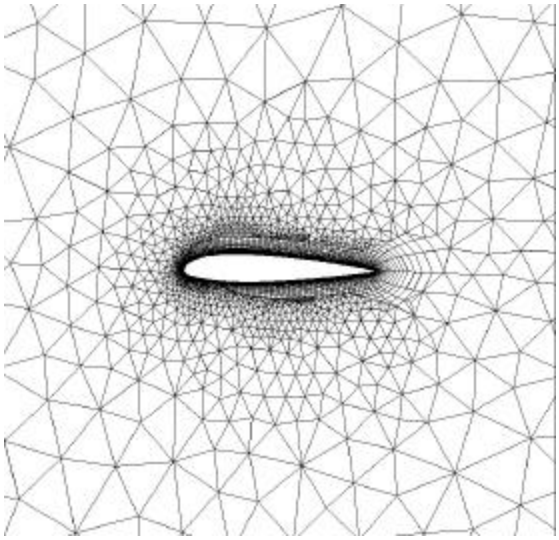


Figure 4 Hybrid Grid Around an Iced Airfoil

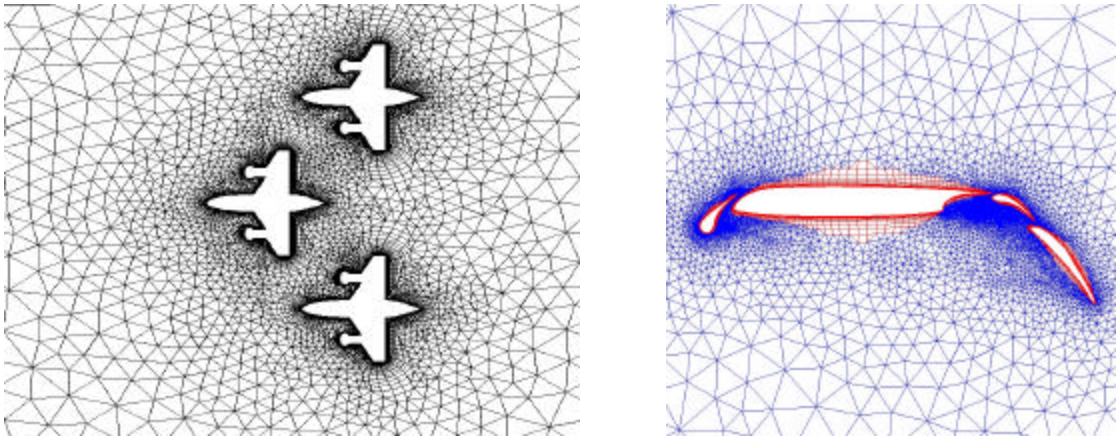


Figure 5 Examples of Hybrid Grids

iced airfoil and the resulting hybrid grid is shown in Figure 4. Two other examples of hybrid grids are shown in Figure 5.

Quad-Dominant Grid Generation

As an alternative to the structured grid method described in the previous section, a quad-dominant, semistructured grid algorithm was developed. The grids generated using the

quad-dominant method can be classified as hybrid grids since they combine elements of both structured and unstructured topologies. They can best be described as semistructured. The marching scheme employed here is similar to the one described in a previous section. However, a node deletion/insertion algorithm is used after the grid is defined for a layer and before the initial surface for the next layer is defined. Using the criteria defined by the algorithm, a point deletion/insertion stencil is created that maintains the structured nature of the grid within layers. This approach creates points that exist in one layer but do not exist in an adjacent layer in so far as the grid generation algorithm is concerned. Traditionally, these grid points have been called hanging nodes. Details regarding the algorithm are given in Reference 6. A simple example illustrating the topology of the grid follows.

Example - Node Deletion at Layer Interface

The node deletion process is illustrated for a two-dimensional grid in Figure 6. The points 1, 2, and 3 form the initial data surface for layer *k*. Points 4, 5, and 6 are generated using the parabolic marching algorithm. Note that points 4, 5, and 6 are not collinear. Based on the deletion/insertion algorithm, it is determined that point 5 should not be used to generate the grid in the next layer *k+1*. Therefore, the initial data surface for layer *k+1* contains the points 4 and 6 but not point 5. The marching algorithm is then used to generate the surface containing points 7 and 8. Note, however, that the cell in layer *k+1* is actually defined using points 4, 5, 6, 8, and 7 resulting in a five-sided cell even though point 5 was not used to compute points 7 and 8. The extension to three dimensions occurs through the deletion of the line passing through point 5 rather than point 5 alone. As an example of this type of grid topology, Figures 7.a and 7.b show a typical grid for an airfoil with ice accretion.

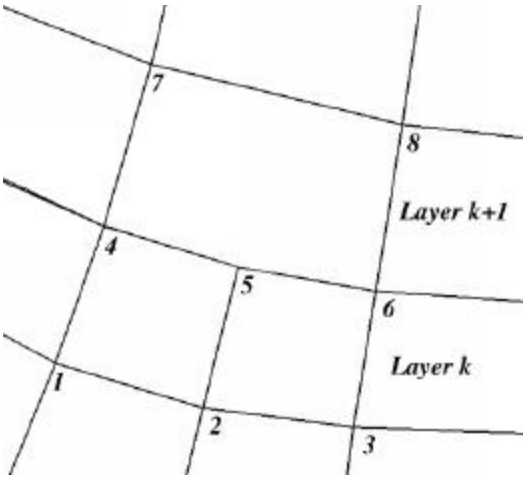


Figure 6 Semistructured Grid Topology

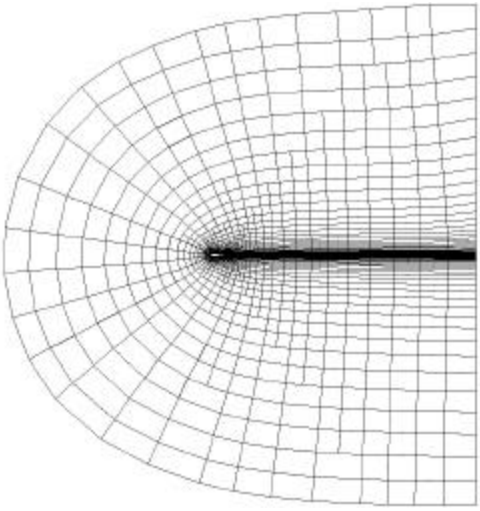


Figure 7.a Overall View of Grid

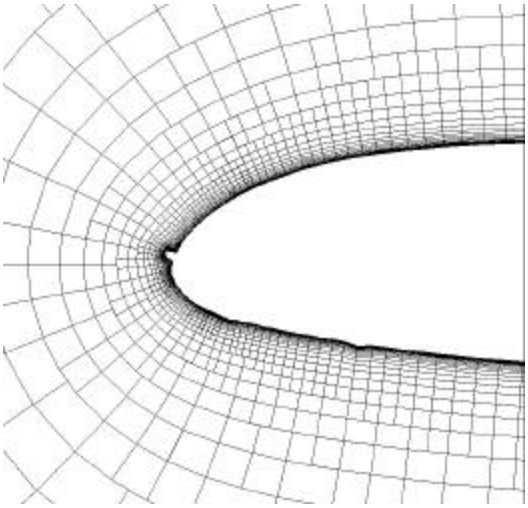


Figure 7.b Detail in Node Region of Airfoil with Ice Accretion

ADAPTATION

Approach to Adaptation

For structured grid topologies, redistribution of a fixed number of grid points, commonly referred to as r-refinement, is by far the simplest to implement and most widely utilized grid adaptation strategy[13]. For unstructured/hybrid or generalized grid topologies where the connectivity is specified explicitly, strategies based on h-refinement, derefinement, and point movement[30] are more attractive and are widely used. Here, redistribution schemes for structured grids with multiple, topologically distinct blocks are explored. Also considered is a point movement approach for generalized grid topologies.

Adaptive redistribution of points traces its roots to the principle of equidistribution of error [37] by which a point distribution is set so as to make the product of the spacing and a positive weight function constant over the set of points:

$$w\Delta x = const.$$

With the point distribution defined by a function x_i, where x varies by a unit increment between points, the equidistribution principle can be expressed as

$$wx_x = const.$$

This one-dimensional equation can be applied in each direction in an alternating fashion [9, 13, 14]. Direct extension to multiple dimensions using algebraic[9], variational [36, 37], and elliptic [13, 14, 17] systems is well documented in the literature. The definition and evaluation of the weight function is the key to a successful adaptation.

Structured grid generation methods using algebraic methods (where interpolation schemes or CAGD (Computer Aided Geometry Design) techniques are utilized) and methods using partial differential equations (where elliptic, parabolic, or hyperbolic equations are solved) can be utilized for adaptive redistribution of grid points. In general, the adaptive redistribution process consists of three main steps. The first step is to define appropriate weight functions representative of important solution features. The second, and probably the most crucial step, is to redistribute the grid points in the computational domain in a manner consistent with the aforestated weight function. It is crucial that the geometric fidelity of solid boundaries must be maintained during the redistribution process. Also, grid quality, as measured by orthogonality, cell aspect ratio, and smoothness, must be maintained. The third and the final step is to modify the metric terms to reflect grid movement with a consistent grid speed or to re-evaluate the flow variables and metric terms at the new grid locations using an appropriate search/interpolation scheme. Time accuracy is achieved by transforming the time derivatives through the addition of convective-like terms containing the grid speeds in a manner that does not alter the conservation properties of the governing PDEs.

The parametric, control points-based, Non-Uniform Rational B-Spline (NURBS) [9, 31] is utilized for algebraic grid adaptation and for redistribution of grid points on solid boundaries associated with the geometrical configuration under study. The convex hull, local support, shape preserving forms, and variation diminishing properties of NURBS are extremely attractive in engineering design applications and in geometry/grid generation. In fact, the NURBS representation has become the "defacto" standard for the geometry description in most modern grid generation systems[28, 31].

Curve/surface/volume grid point redistribution can be accomplished by evaluating the underlying NURBS representation at the specified parametric locations. The usual practice is to transform all pertinent geometrical entities into NURBS representations resulting in a common data structure. However, in many instances, only the grid information is available and the boundaries/surfaces of interest are represented by a discrete set of points. The inverse NURBS formulation can be used to uniquely transform discretized sets or networks of points into appropriate NURBS descriptions.

Weight Functions

Typically, weight functions for solution adaptive grid generation are constructed to mimic the local truncation error and may require substantial user input. These weight functions can then be used to construct blending functions for algebraic redistribution, interpolation functions for unstructured grid generation, forcing functions to attract/repel points in an elliptic system, or to trigger local refinement. Several papers, such as those by Thompson [13], Anderson [14], Soni and Yang [15] and Soni, et al.[16], present applications where significant improvements in accuracy have been obtained through the use of adaptive grid procedures. Most of these techniques are based on weight functions comprised of combinations of flow variable derivatives, and require input for the selection of these variables, as well as their coefficients. Dwyer[17] and Soni and Yang[15] have developed

scaling procedures to lessen the required user input for construction of these weight functions. It is believed that an adaptive grid system which requires no user input while ensuring an efficient grid point distribution would dramatically increase the routine use of CFD in the design and analysis environment. The development of such a system is the ultimate objective of this work. The weight functions developed here utilize extensive scaling and normalization, and hence, may not be the most efficient for a given problem, but are intended to remain useful for a wide range of problems with no user input. The weight function developed here utilizes scaled derivatives and normalizing procedures to minimize or eliminate the need for user input. In this work, solution adaptive redistribution of grid points is achieved via equidistribution of the weight function through the use of forcing functions in elliptic systems, parabolic systems, and algebraic systems.

Construction of Weight Functions

Application of the equidistribution law results in grid spacing inversely proportional to the weight function, and hence, the weight function determines the grid point distribution. Ideally, the weight function would be equal to the local truncation error ensuring a uniform distribution of error. Determination of this function is one of the most challenging areas of adaptive grid generation. The overall solution is only as accurate as the least accurate region. Excessive resolution in a given region does not necessarily increase the accuracy of the overall solution. Evaluation of higher-order derivatives from discrete data is progressively less accurate and is subject to noise. However, lower-order derivatives must be non-zero in regions where the variation of higher-order derivatives is significant, and are proportional to the rate of variation. Therefore, it is possible to employ lower-order derivatives as proxies for the truncation error. Construction of weight functions often requires the user to specify which derivatives are to be used as well as their relative proportions. This can be a time consuming process. Also, due to the disparate strength of flow features, important features can be lost in the noise of dominant features.

Weight functions based on this paradigm have been developed by Soni and Yang[15] and Thornburg and Soni[18]. The weight function of Thornburg and Soni[18] has the attractive feature of requiring no user specified inputs. Relative derivatives are used to detect features of varying intensity, so that weaker but important structures, such as vortices, are accurately reflected in the weight function. In addition, each conservative flow variable is scaled independently. One-sided differences are used at boundaries. No-slip boundaries require special treatment since the velocity is zero and are handled in the same manner as zero velocity regions in the field. A small positive quantity, , is added to all normalizing quantities. In the present work, the weight function uses the Boolean sum construction method of Soni and Yang[15]. Additionally, the normalizing derivatives have been set to an initial or minimum value of ten percent of the freestream quantities. This alleviates problems encountered in flows without significant features to trigger adaption in one or more coordinate directions. Otherwise a few percent variation would be normalized to the same level as a shock or other strong feature.

Analysis of the weight functions explored to date indicates that density or velocity derivatives alone are not adequate to represent the different types and strengths of flow features. Density, or pressure for that manner, varies insufficiently in the boundary layer to be used to construct weight functions for representation of these features. While velocity derivatives by themselves are adequate for resolving boundary layers in viscous flows, additional variables must be included to represent other flow features such as shocks or expansions.

The current weight function is as follows:

$$W = 1 + \frac{W^1}{\max(W^1, W^2, W^3)} \oplus \frac{W^2}{\max(W^1, W^2, W^3)} \oplus \frac{W^3}{\max(W^1, W^2, W^3)}$$

where, k=1,2,3, and

$$W^k = \sum_{\oplus i=1}^{nq} \left\{ \frac{\frac{\left|q^i_{x^k}\right|}{\left|q^i\right|+e}}{\left(\frac{\left|q^i_{x^k}\right|}{\left|q^i\right|+e} \right)_{\max}} \right\} \oplus \left\{ \frac{\frac{\left|q^i_{x^k x^k}\right|}{\left|q^i\right|+e}}{\left(\frac{\left|q^i_{x^k x^k}\right|}{\left|q^i\right|+e} \right)_{\max}} \right\}$$

Usually r, ru, rv, and rw are utilized for i=1, 2, 3, and 4 respectively. However, the user may select variables for adaptation based on the particular application being considered.

The symbol $\oplus$ represents the Boolean sum. Which, for two variables q_1 and q_2, is defined as

$$q_1 \oplus q_2 = q_1 + q_2 - q_1 q_2$$

Note that the Boolean sum is defined only when $0 \le q_i \le 1$, which brings in both logical or/and effects in the evaluation of weights. Normalization and scaling applied in the computation of weight functions ensure the satisfaction of this criteria. Note that the directional weight functions are scaled using a common maximum in order to maintain the relative strength.

Intuition and experience of the user can be used to determine the location of relatively "weak" features such as shear layers. Feature detection could be improved if this "knowledge" could be incorporated into the weight function. It is critical for the adaptive procedure to recognize all flow features and not be dominated by a single feature such as a strong shock. Additionally, only structures that have been at least partially resolved by the flow solver can be detected by the weight function. Any feature completely missed in the simulation will not be detected. Hence the quality of the weight function is dependent upon the quality of the solution. The adaption procedure and the flow solver should be

coupled so that the adapted grid can reflect all the features that are detected as the solution progresses and improves due to adaptation. If the weight functions are to be used directly, such as in the algebraic redistribution technique, smoothing is required. However, when performing approximate equidistribution via forcing functions, it seems to be more effective to smooth the forcing functions directly.

Grid Point Redistribution Methods: Structured Grids

Algebraic Methods

An algebraic interpolation scheme based on transfinite interpolation is widely utilized for grid generation[28, 29]. Transfinite interpolation is accomplished by the appropriate combination of 1D projectors F for the type of interpolation specified. For a three-dimensional interpolation from all six sides (surfaces) of a section, the combination of projectors is the Boolean sum of the three projectors:

$$F_1 \oplus F_2 \oplus F_3 = F_1 + F_2 + F_{31} - F_1F_2 - F_2F_3 - F_3F_1 + F_1F_2F_3$$

For adaptive grid redistribution, a NURBS representation is used as the projector F in each of the directions associated with the transfinite interpolation. Alternatively, a NURBS surface/volume description can be directly evaluated. The NURBS volume is defined by extending the surface definition in a tensor product form and is presented in eqn (7).

Elliptic Methods

The elliptic generation system:

$$\sum_{i=1}^{3}\sum_{j=1}^{3} g^{ij} r_{x^i x^j} + \sum_{k=1}^{3} g^{kk} P_k r_{x^k} = 0$$

where r : Position vector,
g^{ij} : Contravariant metric tensor
x_i : Curvilinear coordinate, and
P_k : Control function.

is widely used for grid generation [3]. Control of the distribution and characteristics of a grid system can be achieved by varying the values of the control functions P_k in eqn. (7) [3]. The application of the one-dimensional form of eqn. (7) combined with equidistribution of the weight function results in the definition of a set of control functions for three dimensions given by:

$$P_i = \frac{(W_i)_{x^i}}{W_i} \qquad i=1,2,3$$

These control functions were generalized by Eiseman [2] as:

$$P_i = \sum_{j=1}^{3} \frac{g^{ij}}{g^{ii}} \frac{(W_i)_{x^i}}{W_i} \qquad i=1,2,3$$

In order to conserve some of the geometrical characteristics of the original grid, the definition of the control functions is extended as:

$$P_i = P_{\text{intialgeometry}} + c_i P_{wt} \qquad i=1,2,3$$

where $P_{initialgeometry}$: Control function based on initial geometry

P_{wt} : Control function based on current solution

x_i : Constant weight factor.

These control functions are evaluated based on the current grid at the adaption step. This can be formulated as:

$$P_i^{(n)} = P_i^{(n-1)} + c_i (P_{wt})^{(n-1)}$$

A flow solution is first obtained with an initial grid. Then the control functions Pi are evaluated in accordance with eqns. (17) and (20), which are based on a combination of the geometry of the current grid and the weight functions associated with the current flow solution[13].

Evaluation of the forcing functions corresponding to the current grid geometry has proven to be troublesome. Direct solution of eqn. (7) for the forcing functions using the input grid coordinates via Cramer's rule or IMSL libraries was not successful. For some grids with very high aspect ratio cells and/or very rapid changes in cell size, the forcing functions became very large. The use of any differencing scheme other than the one used to evaluate the metrics, such as the hybrid upwind scheme[8], would result in very large mesh point movements. An alternative technique for evaluating the forcing functions based on derivatives of the metrics was implemented[3].

$$P_i = \frac{1}{2}\frac{(g_{11})_{x^i x^i}}{g_{11}} + \frac{1}{2}\frac{(g_{22})_{x^i x^i}}{g_{22}} + \frac{1}{2}\frac{(g_{33})_{x^i x^i}}{g_{33}} \qquad i=1,2,3$$

This technique has proven to be somewhat more robust, but research efforts are continuing in this area.

Parabolic Methods

The effectiveness of utilizing systems of partial differential equations to generate solution adaptive node distributions for discrete solutions of field equations is well documented. In addition to the elliptic techniques described in the previous section, hyperbolic systems (Niederdrenk[19], Niederdrenk[20]) have also been used to generate adaptive grids for a variety of configurations.

The two main advantages provided by elliptic systems are 1) the smoothness of the resulting mesh and 2) the ability to control grid point locations. The primary disadvantage is the execution time required to solve the resulting elliptic system. The main advantage of the hyperbolic system is the execution time needed to generate the grid. Hyperbolic

systems can be solved in much less time than elliptic systems - typically one to two orders of magnitude less. However, purely hyperbolic systems do not guarantee the same degree of smoothness because of a lack of dissipation. This shortcoming is remedied through the addition of dissipative terms (Chan[22])

Parabolic systems provide a compromise between these two approaches since grids generated using parabolic methods can be obtained in times competitive with those generated using hyperbolic schemes. In addition, the presence of dissipation helps to ensure a smooth grid point distribution. The approach used here is based on the initial work of Nakamura[21] for static grids, later extended by Noack and Anderson [23] and Noack and Parpia[24] to solution adaptive grids.

The method employed here is essentially the same method used for the semistructured grid generation only without node deletion/insertion. It should be noted that the node deletion/insertion strategy could be based on flow variables as well so the potential exists for a grid adaptation algorithm based on refinement and redistribution. Although only two-dimensional domains are considered here, extension to three dimensions is straight forward and has previously been demonstrated (Noack and Parpia[24]) for a structured grids.

Figure 8 shows an inviscid, supersonic channel flow with a parabolically generated, solution adaptive grid. The grid clearly adapted to the shock at the corner, the reflected shock and some clustering is evident in the expansion.

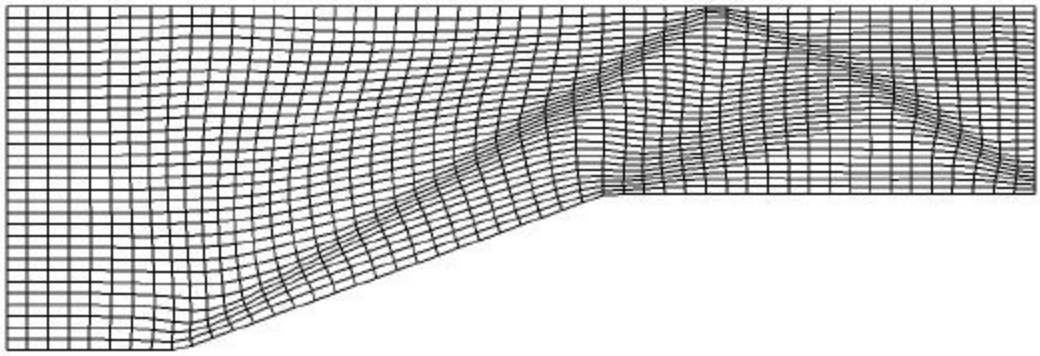


Figure 8. Parabolically Generated, Solution Adapted Grid

GRID POINT MOVEMENT: UNSTRUCTURED GRIDS

One of the many techniques for grid adaptation by node movement is using a weighted Laplacian approach (Thompson et al.[25]). This approach is simple to implement and can be used for any grid topology. The weights are calculated using the same approach as used for structured grids. The gradient at the cell center, used in evaluating the weight function, is calculated by applying Gauss' theorem:

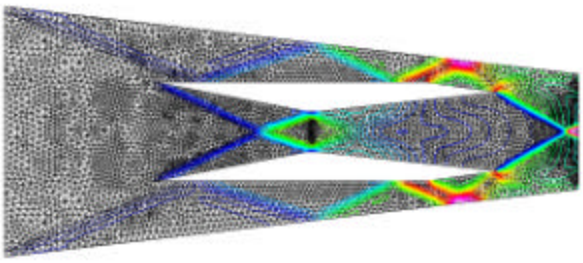
$$\nabla f = \frac{1}{V}\oint_{\partial\Omega} f\,\underline{n}\,dA = \frac{1}{V}\sum_{faces} f\,\underline{n}\,dA$$

where, V is the cell volume, $\underline{n}$ is the positive outward unit normal to the control surface, and dA is the control surface area. The gradients and the corresponding weight functions are calculated in physical coordinates. A typical form of the weighted Laplacian [25] is written as:

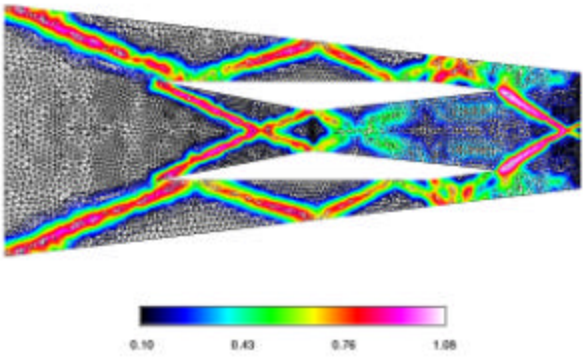
$$\underline{r}^{n+1} = \underline{r}^{n} + w\,\frac{\sum_{edges} W_{i0}\,(\underline{r}_i^n - \underline{r}_0^n)}{\sum_{edges} W_{i0}}$$

where, $\underline{r}$ is the position vector, superscript $n+1$ and n are indicate relaxation levels, W_{i0} is the weight function for the edge connecting nodes i and 0, and ω is the relaxation parameter.

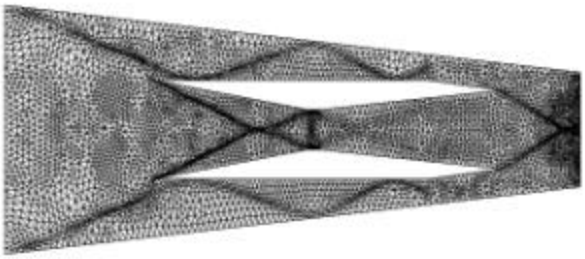
An example of the grid adaptation using the weighted Laplacian approach is shown in Figure 8. An unstructured grid for a geometry representative of a scramjet engine is considered for this example. The inlet Mach number is taken as 3 and the resultant pressure distribution together with initial grid is shown in Figure 8(a). The weight function is based on the four conserved variables and is plotted in Figure 8(b). The adapted grid and the solution on the adapted grid are shown in Figure 8(c) and (d). It can be seen from the pictures that the shocks and expansion fans are captured more distinctly in comparison to the original unadapted grid.



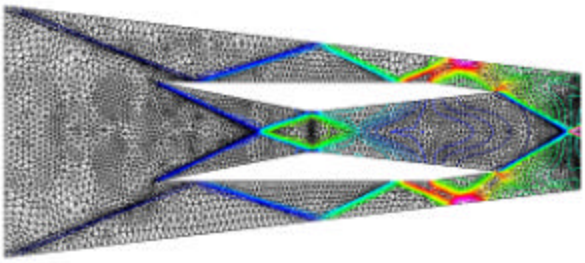
(a) Initial Grid and Pressure Distribution



(b) Weight Function



(c) Adapted Grid



(d) Pressure Distribution on Adapted Grid

Figure 8 Grid Adapatation on Unstructured Grid Using Weighted Laplacian Approach

Surface Point Redistribution

Accurate representation of the flow field in the vicinity of boundaries is critical for an acceptable overall solution. Physical processes occurring near the boundaries often drive the flow physics in other regions of the domain. This is especially true for no-slip surfaces. Hence, the quality and distribution of the grid in no-slip regions is of critical importance. Mesh orthogonality may also required for the implementation of a turbulence model. When using an adaptive procedure based on a redistribution strategy, the interior points move as the grid is adapted. This leads to distorted cells if the boundary points are not redistributed in a consistent manner as the grid is adapted. Both grid quality and geometric fidelity must be maintained during the redistribution process. In this approach, all surfaces of individual blocks are treated in the same manner - whether they are block interfaces or physical boundaries. The NURBS description of the underlying geometry associated with the interface has already been presented. This description is used to

generate the redistributed surface using a user specified distribution mesh. The entire surface or a subregion can be redistributed. Subregions can be used to fix points, such as sharp corners or a transition point between boundary condition type. For solid surfaces, the distribution mesh is based on the nearest interior surfaces. The spacing between surfaces is small and the surfaces are of a similar geometric shape resulting in a nearly orthogonal coordinate system. Block interfaces are treated by redistributing the current block surface based on its corresponding surface in the neighboring block.

Feature Detection

Although the feature detection module described below is designed to be used as a stand-alone software package, the detection technique may also be incorporated into the weight functions used for grid adaptation.

Locating vortices in complex, three-dimensional flow fields can be a challenge. A technique has been developed to automatically locate swirling regions of flow based on the eigenvalues of the velocity gradient tensor[56]. The method is based on a single parameter (termed the swirl parameter) which is defined in regions of complex eigenvalues. For the special case of a spatially and temporally constant velocity gradient tensor (which yields a periodic fluid motion), the magnitude of the complex portion of the eigenvalues can be related to the period of the swirling motion. The swirl parameter is defined to be the ratio of the period implied by the complex portion of the eigenvalues to the time it takes a fluid particle to convect through the region of complex eignvalues. In regions where the swirl parameter is small, the fluid particle convects out of the complex eigenvalue field "too fast" to participate in swirling motion. Where the swirl parameter is large, the fluid swirls before it can be convected out of the region of complex eignevalues. The convection velocity is the velocity in the plane perpendicular to the real eigenvector. For the more realistic case of a spatially and temporally varying velocity gradient tensor, the swirl parameter is interpreted as the tendency for the fluid to swirl around a point.

A software package has been developed that reads grid and q files in Plot3d format and computes the swirl parameter at each point in the field. The data is output in the format of a Plot3d function file. A user-friendly interface has been developed to simplify the process for multiple data files. The interactive version of the program is written in FORTRAN90 and C++ and requires the QT library (http://www.troll.no). Additionally, a batch version of the program has been developed. A pictorial view of the user interface is shown in Figure 9. Two examples of flow visualization using the swirl parameter are shown in Figures 10 and 11. The software is available for DOD users. Additional details can be obtained at http://www.erc.msstate.edu/~dst/research/swirling_flow/swirling_flow.html or via email at dst@erc.msstate.edu.

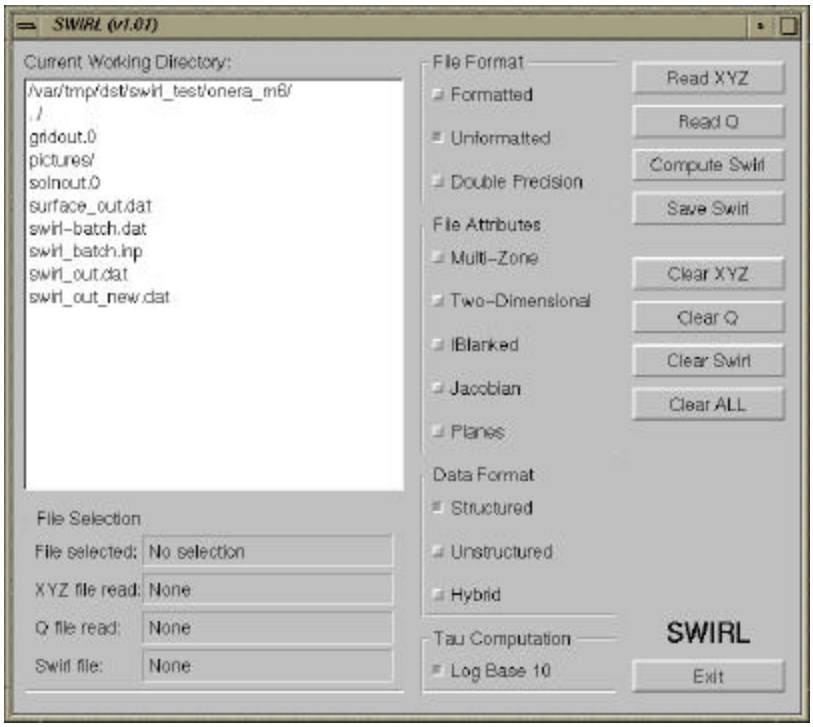


Figure 9. Swirl User Interface

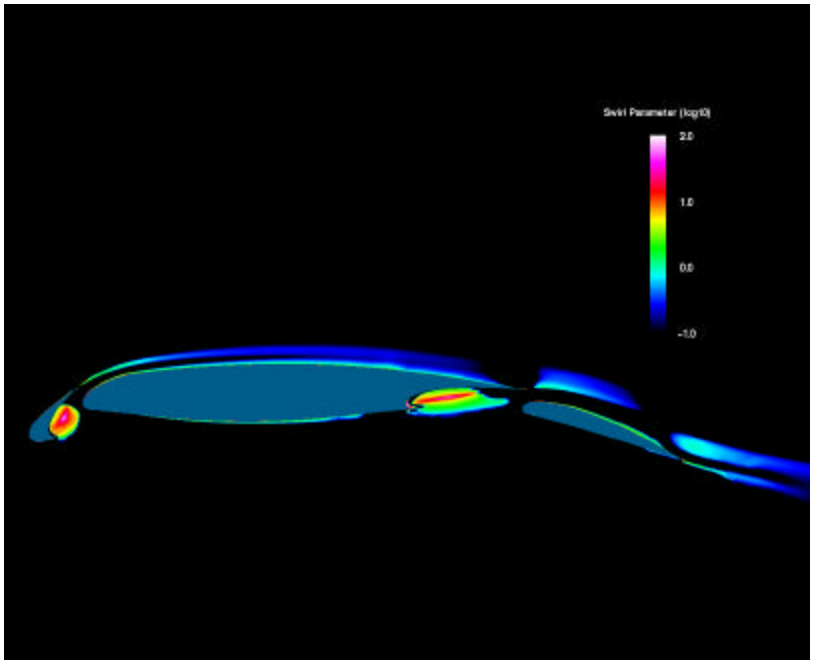


Figure 10. Swirl Parameter for Multi-Element Airfoil

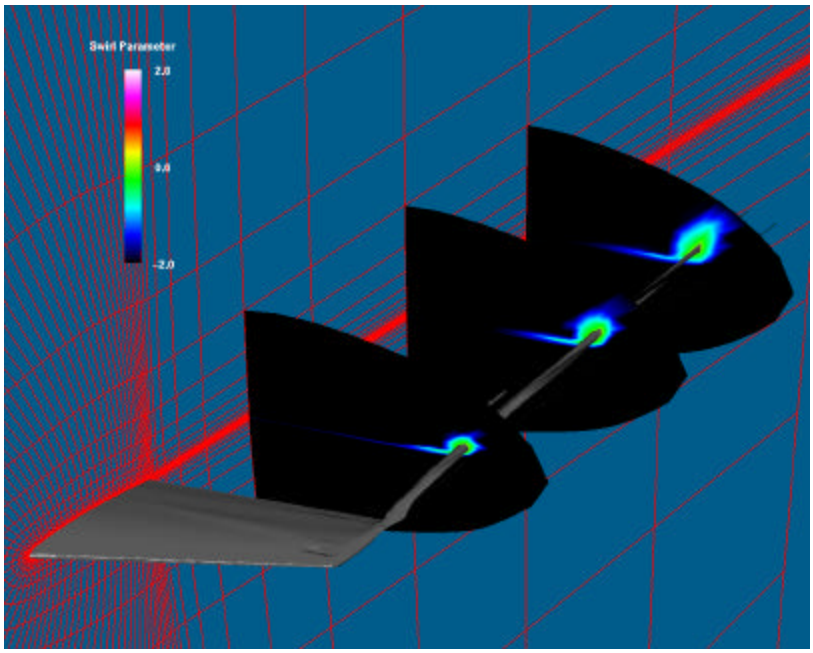


Figure 11. Isolevel Plot of Swirl Parameter for ONERA M6 Wing

PMAG: PARALLEL MULTIBLOCK ADAPTIVE GRID SYSTEM

A parallel grid adaptation system for general three-dimensional structured multiblock domains (PMAG) [34] is described in this section. The grid adaptation scheme is based on the redistribution of grid points using an elliptic solver with weight functions as described previously. The grid blocks are treated as individual domains that may be distributed over multiple processors. MPI [32, 33] is used for message passing. On shared memory machines, each block can be split over multiple threads. The weight function used has been previously described in section 3. A Neumann surface boundary condition has been implemented using NURBS representations for the geometrical entities described as previously to maintain geometric fidelity. A parallel multiblock solution interpolation algorithm has been incorporated to guarantee accurate adaptation. The system can also be used as a multiblock elliptic grid generator. The PMAG system was developed with the following goals in mind:

1. There should be absolutely no restriction on block connectivity. A block can be be connected to any block including itself, thus supporting a wide range of complex three-dimensional topologies.

2. The algorithm should adapt a multiblock grid concurrently with each block solved in an individual process. These processes could be run on a shared memory parallel machine or distributed over a network of workstations. The algorithm should be scalable. Communication latency should be minimized.

3. Grid adaption should require minimum user interaction to resolve all important flow features. The Neumann boundary conditions should be incorporated to maintain grid quality near body surfaces. Solution interpolation must be included.

Parallel Implementation

The distribution of grid points in each block is to be resolved as an independent process. PMAG spawns processes equal to the number of blocks in the topology. Each block is stored in a separate disk file. This is done to enable concurrent reading and writing of grid files by each process. The user needs to supply the connectivity information for each block. This includes information about shared faces and fixed patches. Grid lines must be continuous across adjoining blocks. The inter-block faces, edges, and vertices that do not describe a fixed body must be free to float in the space. This requires that the point distribution on the block faces be computed with an exchange of information across the faces. Each block has a layer of ghost cells that contain data from the neighboring blocks. This data is updated at intermediate intervals using asynchronous communication. Individual processors are responsible for computing the control functions and executing the elliptic solver. A global norm is computed after each iteration for comparison with a specified convergence criterion. Solution interpolation and boundary point movement using a NURBS surface definition is performed after every n iterations as specified by the user.

This algorithm achieves scalability through the use of threads. If excess processors are available, the processes subdivide their domains by unrolling the outermost loop of the solver and the search algorithm and spawning a thread for each subdomain. Controlling the number of threads spawned by each process aids in load balancing. The larger blocks are allowed to spawn more threads than the smaller blocks. Splitting the domain into subdomains also leads to better cache performance. Threads are used with MPI although the MPICH implementation is not thread safe. Therefore, only one thread is active at the time of inter-process communication. The rest of the threads stop during this time. A further enhancement to the current version would have other threads continuing the computations with locks on the data while one thread is dedicated to communications.

Handling Shared Faces

To guarantee complete continuity of grid lines across block faces, each block sends a face to its neighboring block. The elliptic generator can then run using the face from the neighboring block as a Dirichlet boundary. After every iteration, the updated faces are transmitted to the neighboring block. This way, the locations of the face points are computed using the elliptic equations at every step. However, the solution for the face points in each block occurs separately. Hence it is possible that the face points may not coincide after an iteration. To eliminate this discontinuity, some sequential multiblock

codes keep track of the faces, edges, and vertices belonging to each block. Connectivity tables are maintained to identify shared vertices, edges, and faces. At the end of each iteration, these connectivity tables are used to select the copies of the common face, edge, or vertex and then an averaged value (or a value computed using the elliptically solver) is broadcast to all the owners.

Another method especially amenable for C language codes uses pointers to faces, so that only a single copy of a shared face is maintained, with both blocks pointing to the same memory location. Clearly, the later strategy cannot be used when the blocks have been assigned to different processors. The first strategy of collecting all copies and then broadcasting an average can work in a parallel implementation but only with a communication penalty. In the case of a simple structured topology with eight vertices, twelve edges, and six faces per block, each face can be shared by only one other block. One edge is shared by four blocks and a vertex is shared by eight blocks. In such a case, to retrieve unique face, edge, and vertex locations, each block needs to communicate with 26 neighboring blocks. Since an elliptic system has a three-point stencil, and since both blocks have identical copies of all three faces required for computation of the block boundary face, the face points calculated for the each block using the elliptic system should be the same. Hence, the face points are solved individually using point Jacobi iteration (to guarantee a three point stencil).

The point distribution in the interior of the block is computed using the standard tridiagonal system. The control functions for the points on the shared faces are also evaluated using only a three-point stencil guaranteeing that neighboring blocks compute identical locations for the shared points. The simplicity of this scheme is its major advantage. No complicated global connectivity tables need to be maintained. This makes the code extremely flexible, enabling it to handle a wide variety of block topologies, including 0 grids embedded in H grids, periodic boundary conditions, etc. For the above strategy to be successful, the basic premise is that all blocks sharing a particular point must have identical copies of its complete stencil. The user input specifies only the blocks that share a face with the current block. This means that a block has no information about its diagonally opposite neighbor. A communication strategy that allows each block to acquire information from its diagonal neighbors is described in the next section.

Communication between Shared Faces

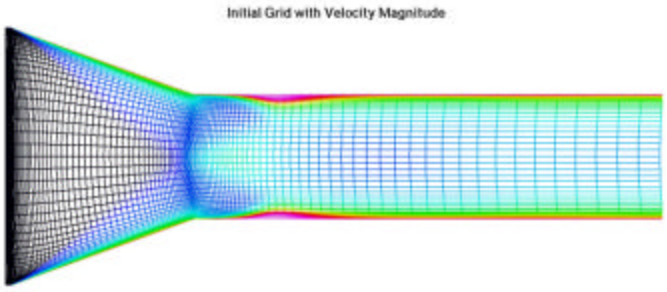
Each block is aware only of the neighboring blocks with shared faces. This means the block cannot directly access the corner points from its diagonally opposite block. To overcome this problem, the blocks communicate faces inclusive of the extra points received from the neighboring blocks (shown by the dashed-blocks). However, this means that the blocks that send the points before receiving the faces from the neighbors would send out old points to the neighbors. One way to overcome this problem is to perform communication in a cyclic loop. However, the resulting communication process would be sequential and result in a large communication latency. The problem can be overcome by simply performing the communication process twice. This strategy allows results in the

use of asynchronous communication, so that more than just two blocks communicate at any instant. In case of the simple topology discussed in the earlier section, each block would now communicate only 12 times. Note that doing the entire communication process twice does involve transmission of some redundant information. This could be avoided by sending only edge and vertex information in the second communication process. This feature has yet to be implemented in PMAG.

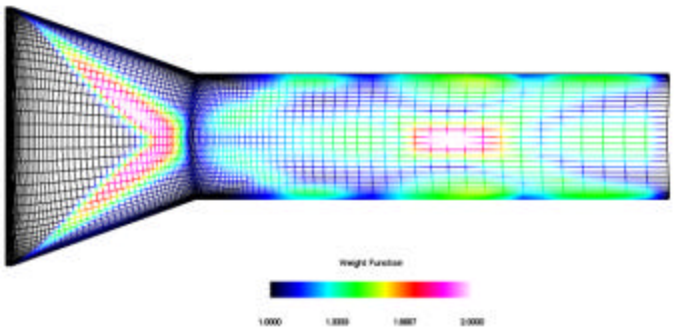
Parallel Multiblock Interpolation

As the points are redistributed, the original solution needs to be interpolated onto the new grid. A parallel grid adaptation algorithm supporting general multiblock topologies makes solution interpolation significantly more complex. The grid points can move outside the original domain of a block. In such a case, the block does not have enough information to interpolate the solution and adaptive functions to all its points. Each block now needs to query all other blocks for the points that are no longer within its domain. The search algorithm starts with a search and interpolation of all points found within each block. Each process creates a list of points that are not found within its domain. These lists are concatenated into a global list which is broadcast to all processes for search. The processes then locate and compute the interpolated solutions for the points found within their domains. A final all-reduce over all processes makes the solutions known to all the processes. This operation takes 4log P communications. A shift operation (the list of external points are shifted in a circle through all processes) would take P communications to complete. Hence a shift would be faster for domains where the number of blocks < 16. Ideally a polyalgorithm should be used to switch methods according to the number of processes. The shift has not yet been implemented in PMAG.

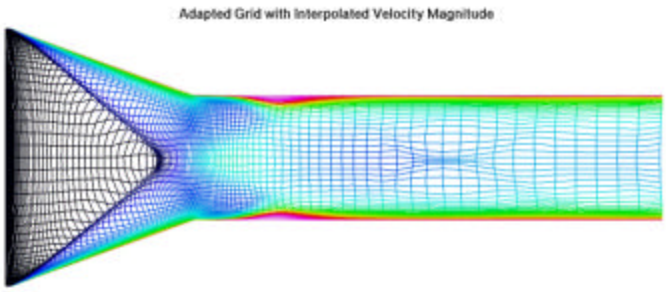
Figure 9 shows a chemically reacting, supersonic flow through a convergent channel. The initial grid and the velocity distribution are shown in Figure 9(a). The adaptation is based on the U, V components of velocity, pressure, temperature and Mach number. Since there was insufficient grid clustering present in the near the shock region in the initial grid, the adaptation was not effective in the vicinity of the shock (see Figure 9(b)). By introducing the metric terms in the weight function, the shock is captured properly (Figure 9(c)) and other solution features are evident in the resulting adapted grid.



(a) Initial Grid and Velocity Distribution



(b) Weight Function



(c) Adapted Grid With Interpolated velocity

Figure 9 Grid Adaptation for Chemically Reacting Flows

The multiblock grid adaptation capability based on the elliptic redistribution scheme is included in the PMAG system. The ability of PMAG to generate grids appropriate for computing complex, three-dimensional flow fields is demonstrated by simulating a supersonic flow around a tangent-ogive cylinder at a Mach number of 1.45 and an angle of attack of 14 degrees [36]. The initial grid and the solution are presented in Figure 11(a)-(b). The weight functions evaluated using the formulation presented in eqns. (10) and (11) is displayed in Figure 11(c). It can be seen that the solution features presented in Figure 11(b) are captured by the weight function. Figure 11(d) shows the adapted grid. The adapted grid shows a concentration of grid points in the primary and secondary shock regions as well as in the boundary layer regions. The concentration in the vortex core is also pronounced

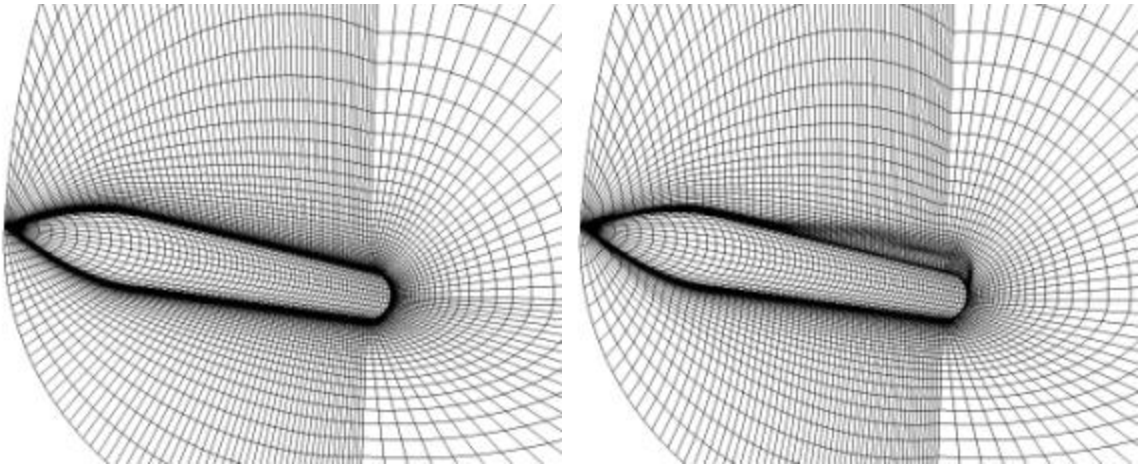
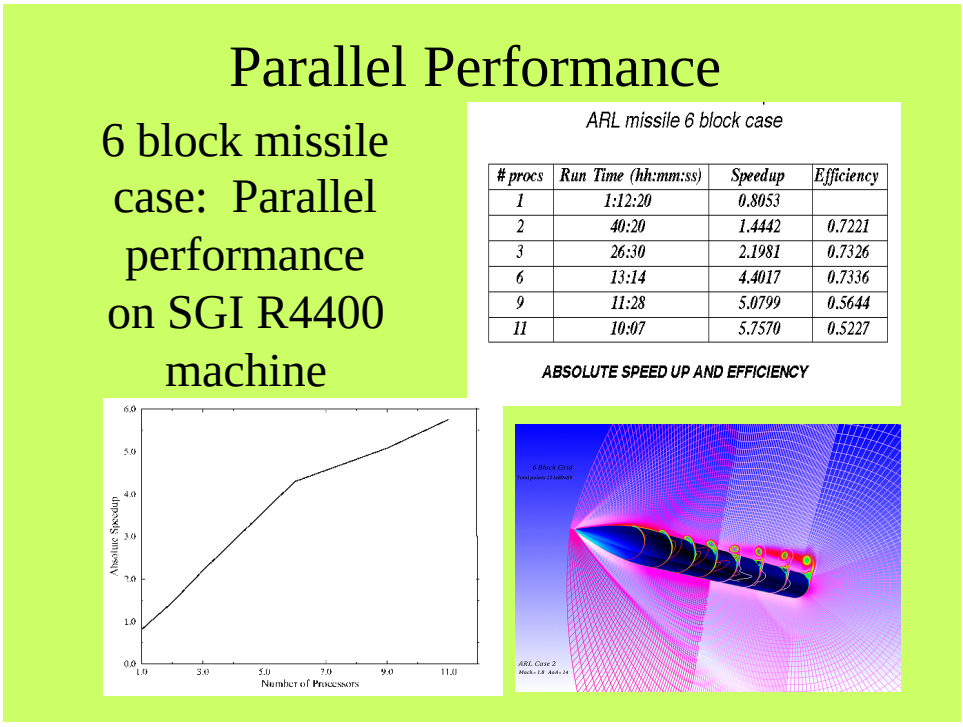


Figure 10 Unadapted and Adapted Grids for KTA2-15 Missile

Figure 12 shows two different longitudinal locations, one displaying the solution and the weight function and the second displaying the solution and the adapted grid. The flow features corresponding to the vortex and the feeding sheet are clearly visible in the weight function as well as in the adapted grid. Axial and normal forces and moments were computed from solutions obtained from the adapted and unadapted grids and are compared with the experimental data in Table 1. The axial force determined from the experimental data is misleading due to a discrepancy during testing[36]. However, the normal force and moment for the adapted grid show better agreement with the experimental data than the normal force and moment for the unadapted grid.

Table 1: Comparison of Forces and Moments

Forces	Experimental Data	Unadapted NPARC (BL)	Adapted NPARC (BL)
Axial	0.1957	0.3307	0.3309
Normal	1.91	1.8855	1.9052
Moment	10.2417	10.0314	10.2060



INTERPOLATION AND SEARCH

The interpolation and search is based on [39]. The algorithm used takes advantage of the local physical properties by mapping the global geometry in terms of local coordinates of the current cell. The interpolation point xp is expressed in terms of these local coordinates which fall within a known interval (between 0 and 1 in case of unstructured and between -

1 and 1 in case of structured) if xp lies within the cell. If any of the coordinates fall outside the interval, the next cell is guessed based on the value of those local coordinates.

In the unstructured search, there are 4 local coordinates.

$$\underline{X}_p = \sum_{j=1}^{4} l_j \underline{X}_j$$

$$\sum_{i=1}^{4} l_i = 1$$

The equations for these are solved directly by inverting using Cramer's rule. The next cell is in the direction of the lowest valued local coordinate. If the next cell is a boundary, then an alternate cell is decided by the direction of the next lowest valued local coordinate.

In the structured search, there are 3 local coordinates.

$$\underline{X}_p = \sum_{j=1}^{8} l_j \underline{X}_j$$

$$l_1 = (1-a)(1-b)(1-g)/8. \qquad l_2 = (1+a)(1-b)(1-g)/8$$

$$l_3 = (1+a)(1+b)(1-g)/8. \qquad l_4 = (1-a)(1+b)(1-g)/8$$

$$l_5 = (1-a)(1-b)(1+g)/8. \qquad l_6 = (1-a)(1+b)(1+g)/8$$

$$l_7 = (1+a)(1+b)(1+g)/8. \qquad l_8 = (1+a)(1-b)(1+g)/8$$

The equations to compute the local coordinates are solved using newton's method. This is because solving using Cramer's rule may give a direct solution that can give a wrong direction sense. In structured search, the next guess cell can be diagonal to the current cell if more than one coordinates fall outside the interval. Further if any of the coordinate is extremely large then the algorithm intelligently jumps more than one cell in the appropriate direction. If we assume the nearby cells to have approximately similar aspect ratio as the current cell, then an a=10 would mean that the point xp lies approximately 10 cells away.

SUMMARY

In this paper, we have described efforts by the CFD CTA to support DOD CFS users through technology improvement projects in grid generation. Additional information can be obtained by contacting Bharat Soni at (601) 325-2647 or bsoni@erc.msstate.edu.

REFERENCES

1 Huang, Chih-Ti., “Hybrid Grid Generation System,” Master's Thesis, Department of Aerospace Engineering, Mississippi State University, MS, August 1996.

2 Weatherill, N. P., “Grid Generation by Delaunay Triangulation,” Lecture Series in von Karman Institute for Fluid Dynamics 1993-94.

3 Thompson, J.F., Warsi, Z.U.A. and Mastin, C.W., Numerical Grid Generation: Foundations and Applications, North-Holland, Amsterdam. 1985.

4 Eiseman, P.R., "Alternating Direction Adaptive Grid Generation," AIAA Paper 83-1937, 1983.

5 Soni, B.K., "Structured Grid Generation in Computational Fluid Dynamics," Vichnevetsky, R., Knight, D., and Richter, G. (eds.), Advances in Computer Methods for Partial Differential Equations VII, Rutgers University, pp 689-695, June 1992.

6 Buning, P.G. and Steger, J.L., "Graphics and Flow Visualization in Computational Fluid Dynamics," AIAA Paper 85-1507-CP, Proceedings of the AIAA 7th Computational Fluid Dynamics Conference, 1985.

7 NASA LeRC and USAF AEDC, "NPARC 1.0 User Notes," June 1993.

8 Ghia, K.N., Ghia, U., Shin, C.T. and Reddy, D.R., "Multigrid Simulation of Asymptotic Curved-Duct Flows Using a Semi-Implicit Numerical Technique," Computers in Flow Prediction and Fluid Dynamics Experiments, ASME Publication, New York 1981.

9 Soni, B.K. and Yang, J.C., "General Purpose Adaptive Grid Generation System," AIAA-92-0664, 30th Aerospace Sciences Meeting, Reno, NV, Jan. 6-9, 1992.

10 Thornburg, H.J. and Soni, B.K., "Weight Functions inGrid Adaption," Proceedings of the 4th International Conference in Numerical Grid Generation in Computational Fluid Dynamics and Related Fields held at Swansea, Wales 6-8th April 1994.

11 Soni, B.K., Thompson, J.F., Stokes, M.L. and Shih, M.H., "GENIE++, EAGLEView and TIGER: General and Special Purpose Interactive Grid Systems," AIAA-92-0071, 30th Aerospace Sciences Meeting, Reno, NV, Jan. 6-9, 1992.

12 NASA LaRC, "User Document for CFL3D/CFL3DE (Version 1.0)", 1993.

13 Thompson, J.F., "A Survey of Dynamically-Adaptive Grids in Numerical Solution of Partial Differential Equations," Applied Numerical Mathematics, vol. 1, pp 3-27, 1985.

14 Anderson, D.A., "Adaptive Grid Methods for Partial Differential Equations," Advances in Grid Generation, ASME Publication, New York, pp. 1-15, 1983.

15 Soni, B.K. and Yang, J.C., "General Purpose Adaptive Grid Generation System," AIAA-92-0664, 30th Aerospace Sciences Meeting, Reno, NV, Jan. 6-9, 1992.

16 Soni, B.K., Weatherill, N.P. and Thompson, J.F., "Grid Adaptive Strategies in CFD," International Conference on Hydo Science and Engineering, Washington, D.C., June 7-11, 1993.

17 Dwyer, H.A., "Grid Adaption for Problems in Fluid Dynamics," AIAA Journal, vol. 22. No. 12, pp. 1705-1712, December 1984.

18 Thornburg, H.J. and Soni, B.K., "Weight Functions inGrid Adaption," Proceedings of the 4th International Conference in Numerical Grid Generation in Computational Fluid Dynamics and Related Fields held at Swansea, Wales 6-8th April 1994.

19 Niederdrenk, P., "Solution Adaptive Grid Generation by Hyperbolic/Parabolized P.D.E.s," Numerical Grid Generation in Computational Fluid Dynamics and Related Fields, Eds. A. S.-Arcilla, J. Hauser, P. R. Eiseman, and J. F. Thompson, Elsevier Science Publishing Company, New York, 1991.

20 Niederdrenk, P., "Grid Adaption to Multiple Auto-Scaled Solution Features," Numerical Grid Generation in Computational Fluid Dynamics and Related Fields, Eds. N.P. Weatherill, P. R. Eiseman, J. Hauser, and J. F. Thompson, Pineridge Press Ltd., Swansea, U.K., 1994.

21 Nakamura, S., "Noniterative Grid Generation Using Parabolic Partial Differential Equations", Numerical Grid Generation, Ed. J. F. Thompson, Elsevier Science Publishing Company, Inc., New York, 1982.

22 Chan, W., "Enhancements of a Three-Dimensional Hyperbolic Grid Generation Scheme," Applied Mathematics and Computation, 51, 181-205, (1992).

23 Noack, R. W., and Anderson, D. A., "Solution-Adaptive Grid Generation Using Parabolic Partial Differential Equations", AIAA J., 28, 1016-1023 (1990).

24 Noack, R. W., and Parpia, I. H., "Solution Adaptive Parabolic Grid Generation in Two and Three Dimensions", Numerical Grid Generation in Computational Fluid Dynamics and Related Fields, Eds. A. S.-Arcilla, J. Hauser, P. R. Eiseman, and J. F. Thompson, Elsevier Science Publishing Company, New York, 1991.

25 Thompson, J. F., Soni, B. K., and Weatherill, N. P., ``Handbook of Grid Generation", CRC Press, 1999, pp. I-26.

26 Yang, J. C., ``General Purpose Adaptive Grid Generation System", Ph.D Dissertation, Dept. of Computational Engineering, Mississippi State University, July 1993.

27 Tleimat, H., ``Computational Simulation of High-Energy Heavy-Ion Collisions", Ph.D Dissertation, Dept. of Physics and Astronomy, December 1996.

28 HandBook of Grid Generation, Thompson, J. F., Soni, B. K., Weatherill, N. P., (Eds), CRC Press, 1999.

29 Soni, B. K., Weatherill, N. P., ``Geometry-Grid Generation", Computer Science and Engineering Hand Book, Allen B. Tucker (Ed), CRC Press, pp 791-819, 1997.

30 Weatherill N. P., ``Mixed Structured-Unstructured Meshes for Aerodynamic Flow Simulation", The Aeronautical Journal, Vol. 94, pp 111-123, No. 934, 1990.

31 Yu, Tzu-Yi., ``CAGI: Computer Aided Grid Interface", Ph.D Dissertation, Dept. of Computational Engineering, Mississippi State University, 1996.

32 Snir, M., Otto, S. W., Huss-Ledermann, Walker, S., and Dongarra, J., ``MPI: The Complete Reference", MIT Press, 1996.

33 Gropp, W., Lusk, E., and Skjellum, A., ``Using MPI: Portable Parallel Programming with the Message-Passing Interface", MIT Press, 1994.

34 Apte, M. S., ``Parallel Adaptive Grid Generation for Structured Multiblock Domain", Master's Theses, Dept. of Computational Engineering, Mississippi State University, 1997.

35 McRae, S. D., and Laflin, K. R., ``Dynamic Grid Adaptation and Grid Quality", HandBook of Grid Generation, Thompson, J. F., Soni, B. K., Weatherill, N. P., (Eds), CRC Press, 1999.

36 Khairullina, O. B., Sidorov, A. F., and Ushakova, O. V., ``Variational Methods of Construction of Optimal Grids", HandBook of Grid Generation, Thompson, J. F., Soni, B. K., Weatherill, N. P., (Eds), CRC Press, 1999.

37 Brackbill, J. S., ``Application and Generalization of Variational Methods for Generating Adaptive Meshes", Numerical Grid Generation, Edited by J. F. Thompson, Elsevier Sciences Publishing Company, 1985.

38 Walter, B. S., Trevor, B., Lauzon, Marc, Housh, C., Manter, J., Josyula, E., and Soni, B. K., ``The Application of CFD to the Prediction of Missile Body Vortices", AIAA 97-0637, 35th Aerospace Sciences Meeting & Exhibit, Jan. 6-10, 1997, Reno, NV.

39 Stokes, M. L. and Kneile, K. R., "A Three Dimensional Search/InterpolationScheme for CFD Analysis", Presented at the first World Congress on Computational Mechanics, University of Texas at Austin, September 1986.

40 Melton, J.E., Berger, M.J., Aftosmis, M.A., and Wong, M.J., "3D Applications of a Cartesian Grid Euler Method," AIAA 95-0853, January 1995.

41 M.J. Berger, "Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations," *Journal of Computational Physics*, 53: 484-512, 1984.

42 Gordon, W.J. and Thiel, L.C., "Transfinite Mappings and Their Application to Grid Generation," *Numerical Grid Generation*, Thompson, J.F. (ed.), North Holland, Amsterdam, 1982.

43 Soni, B.K., "Grid Generation for Internal Flow Configurations," *Computer & Mathematics with Applications*, Vol. 24, No. 5/6, pp. 191-201, September 1992.

44 Thompson, J.F., "A General Three-Dimensional Elliptic Grid Generation System on a Composite Block Structure," *Computer Methods in Applied Mechanics and Engineering*, 64, 377-411, North Holland, 1987.

45 Steger, J.L. and Chausee, D.S., "Generation of Body-Fitted Coordinates using Hyperbolic Pratila Differential Equations," *SIAM Journal of Scientific Computation*, p. 431, 1980.

46 Brackbill, J.U., "An Adaptive Grid with Directional Control," *Journal of Computational Physics*, Vol. 108, p. 38, September 1993.

47 Voronoi, G., "Nouvelles Applications des Parametres Continus a La Theorie Des Formes Quadratiques, Rescherches sur les Parallelloedres Primitifs," J. Reine Angew. Math., Vol. 134, 1908.

48 Lohner, R. and Parikh, P., "Three-Dimensional Grid Generation by the Advancing-Front Method," *International Journal of Numerical Methods of Fluids*, 8, 1135-1149, 1988.

49 Marcum, D.L., "Generation of Unstructured Grids for Viscous Flow Applications," AIAA 95-0212, 33rd Aerospace Sciences Meeting and Exhibit, January 9-12, 1995, Reno, NV.

50 Koomullil, R. P., Soni, B.K., and Chih-Ti, H., "Navier-Stokes Simulation on Hybrid Grids," AIAA 96-767, also presented at the 34th Aerospace Sciences Meeting, January 15-18, 1996, Reno, NV.

51 Thompson, J.F., "A Reflection on Grid Generation in the 90's: Trends, Needs, and Influences," *International Numerical Grid Generation in Computational Field*

Simulations, B.K. Soni, J.F. Thompson, J. Hauser and P.R. Eiseman (Eds.), P. 1029, Proceedings of the 5th International Grid Generation Conference, ERC Press, 1996.

52 Soni, B.K., Weatherill, N.P., "Geometry-Grid Generation," *Computer Science and Engineering Handbook*, CRC Press, pp.791-816, 1996.

53 Thompson, J.F., Soni, B.K., Weatherill, N.P., (Eds.), *CRC Handbook of Grid Generation*, CRC Press, 1998.

54 Sahu, J., Heavey, K., and Nietubicz, C. "Computational Modeling of Sense And Destroy Armor (SADARM) Submunition Separation/Collision," ARL-TR-1378, June 1997.

55 Sahu, J., Dinavahi, S., Heavey, K., and Nietubicz, C., "Application of Chimera Grid Technique to Projectile Configurations," presented at 6th International Conference on Numerical Grid Generation in Computational Field Simulation, July 98, University of Greenwich, London, England.

56 C. H. Berdahl and D. S. Thompson, "Eduction of Swirling Structure Using the Velocity Gradient Tensor," AIAA J., Vol. 31, No. 1, 1993, pp. 97-103, (also AIAA Paper 91-1823, presented at the AIAA 22nd Fluid Dynamics, Plasma Dynamics and Lasers Conference, Honolulu, HI, 1991)